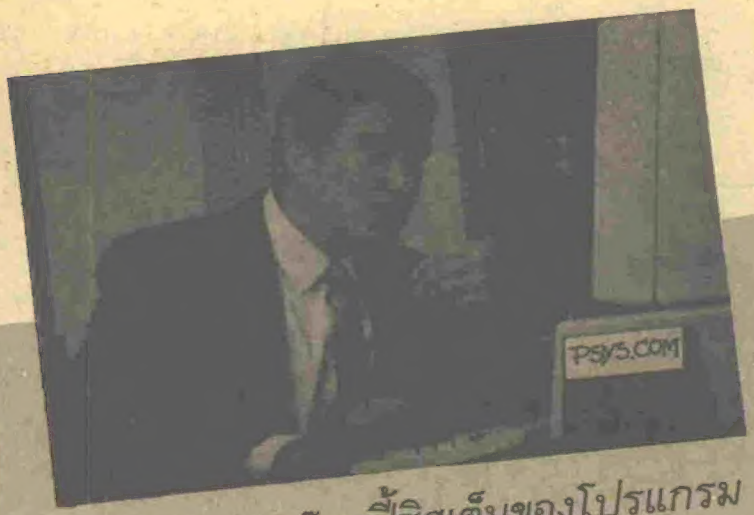
PSYS.COM
โปรแกรมช่วยก๊อบปี้ซิสเต็มสำหรับแผ่นข้อมูล



แก้ปัญหาการก๊อบปี้ซิสเต็มของโปรแกรม SYS.COM ด้วย PSYS.COM ที่ช่วยจัดเนื้อที่สำหรับแผ่นข้อมูล เมื่อต้องการให้เป็นแผ่นข้อมูลที่มีซิสเต็ม

ขจร โรจนเมธินทร์

ทุกครั้งที่คุณนำดิสค์แผ่นใหม่มาใช้งานคุณจะต้องตัดสินใจว่าจะฟอร์แมตดิสค์แผ่นนั้นให้มีซิสเต็มไฟล์หรือไม่ ซึ่งถ้าคุณเลือกแบบมีซิสเต็มไฟล์ คุณก็จะเสียเนื้อที่บนดิสค์สำหรับเก็บไฟล์ IBMBIO.COM, IBMDOS.COM, COMMAND.COM ไปไม่น้อยกว่า 68 กิโลไบต์ อย่างไรก็ตามการมีซิสเต็มไฟล์บนดิสค์ทำให้มีความสะดวกในการใช้งานคือ คุณสามารถใช้ดิสค์แผ่นนั้นบูตเครื่องได้ และช่วยให้ไม่ต้องเปลี่ยนดิสค์ใหม่เมื่อเครื่องต้องการอ่านส่วน Transient ของ COMMAND.COM ลงบนหน่วยความจำ

คราวนี้หากคุณเลือกฟอร์แมตแบบไม่มีซิสเต็มไฟล์ แล้วคุณเกิดเปลี่ยนใจอยากให้ดิสค์แผ่นนั้นมีซิสเต็มไฟล์ คุณอาจใช้โปรแกรม SYS.COM ของดอสช่วยเพิ่มซิสเต็มไฟล์ลงบนดิสค์ได้ แต่ SYS.COM จะเพิ่มซิสเต็มไฟล์ได้ในกรณีที่ดิสค์แผ่นนั้นยังไม่มีข้อมูลเก็บอยู่เท่านั้น โปรแกรม PSYS.COM จะช่วยย้ายข้อมูลบนดิสค์ใหม่โดยทำให้เกิดที่ว่างพอสำหรับซิสเต็มไฟล์ จากนั้นคุณก็สามารถใช้ SYS.COM เพิ่มซิสเต็มไฟล์ได้

```
PSYS B:
IF ERRORLEVEL 1 GOTO END
SYS B:
COPY COMMAND.COM B:
:END
```

รูปที่ 1 แสดงคำสั่งในแบตช์ไฟล์

แนะนำตัว PSYS.COM

PSYS.COM จะจัดเนื้อที่บนดิสค์ให้เกิดที่ว่างสำหรับซิสเต็มไฟล์โดยจะตรวจสอบที่ว่างบนดิสค์ก่อนว่ามีเหลือพอสำหรับซิสเต็มไฟล์หรือไม่ หากไม่พอก็จะยกเลิกการทำงานและถ้าดิสค์แผ่นนั้นมีซิสเต็มไฟล์อยู่แล้วคุณก็สามารถลบซิสเต็มไฟล์เก่าได้ ในขณะที่ PSYS.COM ทำงานก็จะตรวจข้อผิดพลาดต่าง ๆ ที่อาจมีบนดิสค์ของคุณ เช่น มีเซกเตอร์เสีย, มีข้อผิดพลาดใน File Allocation Table (FAT) เป็นต้น หากพบข้อผิดพลาด PSYS.COM ก็จะรายงานให้ทราบทันที

หลังจากที่ PSYS.COM ทำงานเรียบร้อยแล้ว คุณจะต้องใช้ SYS.COM เพิ่มซิสเต็มไฟล์ต่อทันที หากคุณพยายามที่จะใช้ดิสค์แผ่นนั้นก่อนจะเพิ่มซิสเต็มไฟล์ดอสจะคิดว่าคุณกำลังใช้ดิสค์แผ่นใหม่ซึ่งยังไม่มีข้อมูล ทั้งนี้เพราะ PSYS.COM ทำการแก้ไข FAT และดิสค์ไดเรคทอรีใหม่เพื่อเตรียมให้ SYS.COM ทำงานต่อและเมื่อ SYS.COM ทำงานเสร็จ ขั้นต่อไปคุณก็ต้องก๊อบปี้ COMMAND.COM มาไว้บนดิสค์แผ่นนั้นด้วย คุณก็จะได้ดิสค์ที่มีซิสเต็มมาใช้งานทันที

เพื่อความสะดวกคุณอาจสร้างแบตช์ไฟล์ขึ้นมาให้ทำหน้าที่ตามขั้นตอนเหล่านี้ดังรูปที่ 1 ทั้งนี้เพราะ PSYS.COM จะคืนค่าให้พารามิเตอร์ ERRORLEVEL ซึ่งถ้ามีข้อผิดพลาดเกิด พารามิเตอร์ ERRORLEVEL จะมีค่าเป็น 1

โปรแกรม PSYS.COM ถูกเขียนให้ทำงานได้กับฟลอปปี้ดิสค์ขนาด 5.25

นิ้ว แบบ 40 แทร็ก แต่ละแทร็กแบ่งออกเป็น 8 หรือ 9 เซกเตอร์ต่อแทร็ก เก็บได้ 1 หรือ 2 หน้าเท่านั้น หากคุณใช้ PSYS.COM กับดิสค์แบบอื่น โปรแกรมจะตรวจพบข้อผิดพลาด และยกเลิกการทำงานทันที (ตัวโปรแกรมดังรูปที่ 2)

ข้อจำกัดอื่น ๆ สำหรับ PSYS.COM ก็คือ ไม่ควรใช้ PSYS.COM กับดิสค์ที่มีสับไดเรคทอรี และไม่ควรใช้ PSYS.COM กับดิสค์ที่ป้องกันการก๊อบปี้ เนื่องจากดิสค์เหล่านี้มักจะมีฟอร์แมตที่แตกต่างไปจากฟอร์แมตของดอส

มารู้จักกับ File Management ของดอส

หน้าที่สำคัญอันหนึ่งของระบบปฏิบัติการก็คือ การจัดระเบียบของข้อมูลบนดิสค์ โดยจัดเนื้อที่ให้แต่ละไฟล์ไม่ซ้ำกัน ซึ่งผมจะขอกล่าวถึงเฉพาะฟลอปปี้ดิสค์เท่านั้น

ดอสแบ่งชนิดของฟลอปปี้ดิสค์ออกเป็น 4 ชนิดดังนี้คือ

1. เก็บได้ 1 หน้า และแต่ละแทร็กแบ่งออกเป็น 8 เซกเตอร์
2. เก็บได้ 2 หน้า และแต่ละแทร็กแบ่งออกเป็น 8 เซกเตอร์
3. เก็บได้ 1 หน้า และแต่ละแทร็กแบ่งออกเป็น 9 เซกเตอร์
4. เก็บได้ 2 หน้า และแต่ละแทร็กแบ่งออกเป็น 9 เซกเตอร์

ดิสค์แต่ละแผ่นจะมี 40 แทร็ก และเมื่อเราฟอร์แมตดิสค์แต่ละแทร็กก็จะแบ่งออกเป็นเซกเตอร์ขนาด 512 ไบต์ ดังนั้นดิสค์แบบที่ 4 ก็จะมี 720 เซกเตอร์ และมีความจุ 360 กิโลไบต์ โดยดอสจะแบ่งพื้นที่บนดิสค์เพื่อใช้เก็บบูตเซกเตอร์ 1 เซกเตอร์, ดิสค์ไดเรคทอรี 7 เซกเตอร์, FAT 4 เซกเตอร์ (โดยเก็บ

รูปที่ 2
โปรแกรม PSYS.ASM

```
code            segment para public 'code'
                assume cs:code,ds:code,es:code
                org 100h
begin:          jmp prep
;
copy_right      db 13,10,'Copyright 1989 KJ''s software',13,10,'$'
errmsg1         db 13,10,'Invalid Drive Specifier',13,10,'$'
errmsg2         db 13,10,'Illegal Disk Format',13,10,'$'
errmsg3         db 13,10,'Disk Read Error',13,10,'$'
errmsg4         db 13,10,'Disk Write Error',13,10,'$'
errmsg5         db 13,10,'Insufficient Space in Disk Directory',13,10,'$'
errmsg6         db 13,10,'Insufficient Disk Space',13,10,'$'
errmsg7         db 13,10,'Allocation Chain Error',13,10,'$'
errmsg8         db 13,10,'Cannot Delete System Files',13,10,'$'
errmsg9         db 13,10,'Bad Sector in System Area',13,10,'$'
msg1      db 13,10,'System Perp Completed',13,10,'$'
msg2      db 13,10,'System Already Installed - Delete Old System (Y/N)?',13,10,'$'
msg3      db 13,10,'Execution Aborted',13,10,'$'
;
last_cluster        dw 41               ;last available cluster found
sec_per_cluster     dw ?                ;number of sector per cluster
clusters            dw ?                ;total number of data clusters on disk
DIR_entries         dw ?                ;maximum number of directory entries
DIR_sectors         dw ?                ;number of sectors used for directory
FAT_sectors         dw ?                ;number of sectors used for one copy of the FAT
FAT_entry           dw ?                ;temporary storage for any FAT entry
drive               db ?                ;current disk drive
format_table        dw 64,4,2           ;S-9 disk format parameter
                    dw 112,7,2          ;D-9   "     "
                    dw 64,4,1           ;S-8   "     "
                    dw 112,7,1          ;D-8   "     "
system_id       db 'IBMBIO',32,32,'COM' ;text of system file IBMBIO.COM
ibmbio          db 'B:IBMBIO.COM',0     ;ASCIIZ string for DOS file functions
ibmdos          db 'B:IBMDOS.COM',0     ;  "     "     "
cbuffer         dw offset endprog       ;pointer to buffer area for cluster transfer
FAT_data        dw offset endprog+1024  ;pointer to area where FAT image is stored
DIR_data        dw offset endprog+2048  ;pointer to area where DIR image is stored
;
prep            proc near               ;start of main program
;
;Parse the command line for a drive specifier. If none, use default drive.
;
                cld                     ;clear DF for string operation
                mov si,80h              ;point SI to start of command line
                lodsb                   ;get number of characters entered
```

```
46              cmp al,1                ;one or less characters specified?
47              jbe prep1               ;yes, then use default drive
48              inc si                  ;skip first delimeter
49              lodsb                   ;get drive specifier
50              and al,0DFh             ;capitalize it
51              sub al,'A'              ;convert ASCII to drive number
52              cmp al,1                ;is drive specified A: or B:?
53              jbe prep2               ;yes, then finish
54              lea dx,errmsg1          ;no, then error
55              jmp error_exit          ;display message and abort
56 prep1:       mov ah,19h              ;get current drive function
57              int 21h
58              cmp al,1                ;is it drive A: or B:?
59              jbe prep2               ;yes, skip error routine
60              lea dx,errmsg1

61 ;
62 ;Now program execution is routed here when a critical error is detected.
63 ;
64 error_exit:  mov ah,9                 ;print string function
65              int 21h
66              mov ah,2                ;print a character
67              mov dl,7                ;beep
68              int 21h
69 error1:      mov ah,4Ch              ;terminate program function
70              mov al,1                ;set ERRORLEVEL return code to 1
71              int 21h
72 ;
73 ;Check the format of the disk to verify that it's a 40-track diskette
74 ;(8 or 9 sectors per track, one or two sides).
75 ;Store information pertaining to the disk format.
76 ;
77 prep2:      mov drive,al            ;save current drive ID
78              mov dl,al               ;get drive in DL
79              inc dl                  ;adjust drive ID for DOS function call
80              push ds                 ;save segmetn register
81              mov ah,1Ch              ;get FAT information
82              int 21h
83              mov cl,[bx]             ;store ID byte in CL
84              pop ds                  ;restore DS
85              cmp cl,0FCh             ;ID byte greater than or equal to FCh?
86              jae prep2a              ;yes, then format is ok
87              lea dx,errmsg2          ;no, then format is invalid
88              jmp error_exit          ;abort on error
89 prep2a:      cbw                     ;convert byte in AL to word in AX
90              mov sec_per_cluster,ax  ;store number of sectors per cluster
91              mov clusters,dx         ;store total number of data clusters
92              sub cl,0FCh             ;normalize ID byte by subtracting FCh
93              mov al,cl
94              mov dl,6
95              mul dl                  ;multiply by 6
96              mov bl,al               ;transfer table index in AL to BL
97              xor bh,bh               ;convert byte to word
98              add bx,offset format_table    ;form address of disk format parameter line
99              mov ax,[bx]             ;read number of dir entries from table
100             mov DIR_entries,ax      ;save number of entries
101             mov ax,[bx+2]           ;read directory length from table
102             mov DIR_sectors,ax      ;save directory length
103             mov ax,[bx+4]           ;read FAT length from table
104             mov FAT_sectors,ax      ;save FAT length
105 ;
106 ;Read root directory and one copy of FAT into memory from the selected drive.
107 ;
108 prep3:      call read_FAT           ;read in one copy of the file allocation table
109             jnc prep3a              ;continue if no error on read
110             lea dx,errmsg3          ;exit if error occured from reading
111             jmp error_exit
112 prep3a:     call read_DIR           ;read in disk directory
113             jnc prep4               ;continue if no error
114             lea dx,errmsg3
115             jmp error_exit
116 ;
117 ;Determine if the disk is already system formatted by seeing if the first
118 ;file in the directory is IBMBIO.COM.
119 ;
120 prep4:      lea si,system_id        ;point SI to 'IBMBIO.COM'
121             mov di,DIR_data         ;point DI to first filename
122             mov cx,11               ;11 bytes to check
123             repe cmpsb              ;compare the two string
124             jne prep6               ;skip delete routine if no system
125 ;
126 ;Disk already system formatted--see if user wants to delete the system file.
127 ;
128             lea dx,msg2             ;get query message address
129             mov ah,9                ;print string function
130             int 21h
131 prep5:      mov ah,0                ;get key press
132             int 16h
133             and al,0DFh             ;capitalize it
134             cmp al,'Y'              ;was 'Y' pressed?
135             je prep5a               ;yes, delete system file
136             cmp al,'N'              ;was 'N' pressed?
137             jne prep5               ;no, then get another keypress
138             lea dx,msg3             ;get abort message addres
139             mov ah,9                ;print message
140             int 21h
141             jmp error1              ;exit with errorlevel 1
142 ;
143 ;Delete system files - change file attributes to 0 (eliminating read-only bit)
144 ;before using DOS delete function.
145 ;
146 prep5a:     cmp drive,1             ;is B: the current drive?
147             je prep5b               ;yes, continue
148             dec ibmbio              ;no, set ASCIIZ drive designator to A:
149             dec ibmdos
```

```
150 prep5b:     lea dx,ibmbio           ;change IBMBIO.COM file attribute for deletion
151             mov ah,43h              ;change file attribute
152             mov cx,0                ;new file attribute
153             mov al,1                ;specify that attribute is to be changed
154             int 21h
155             jc prep5c               ;exit on error if unsuccessful
156             lea dx,ibmdos           ;do the same
157             mov ah,43h
158             mov cx,0
159             mov al,1
160             int 21h
161             jc prep5c
162             lea dx,ibmbio           ;now delete IBMBIO.COM
163             mov ah,41h              ;delete file function
164             int 21h
165             jc prep5c
166             lea dx,ibmdos
167             mov ah,41h
168             int 21h
169             jc prep5c
170             jmp prep3               ;reread FAT and directory
171 prep5c:     lea dx,errmsg8          ;files cannot be deleted
172             jmp error_exit
173 ;
174 ;Check that the disk has adequate free space before continuing.
175 ;
176 prep6:      mov dl,drive            ;get drive in DL
177             inc dl                  ;adjust drive ID for DOS function call
178             mov ah,36h              ;get free space function
179             int 21h
180             mov ax,sec_per_cluster  ;get sector per cluster
181             mul bx                  ;multuply to get free sectors
182             cmp ax,100              ;at least 100 sectors free?
183             jae prep7               ;yes, then continue
184             lea dx,errmsg6          ;no, then load error message
185             jmp error_exit
186 ;
187 ;Check first two directory entries to see whether each is used or available.
188 ;For each one not available, transfer it to the first available space.
189 ;
190 prep7:      mov si,DIR_data         ;point SI to directory image
191             cmp byte ptr [si],0     ;is the first entry available?
192             je prep9                ;yes, then goto next entry
193             cmp byte ptr [si],0E5h  ;erased file?
194             jne prep7a              ;no, then move it
195             mov byte ptr [si],0     ;mark entry as 'Never used'
196             jmp prep9               ;jump to next entry
197 prep7a:     mov dx,2                ;entry is occupied
198             call find_entry         ;find first available directory entry
199             jnc prep8               ;entry found - continue
200             lea dx,errmsg5          ;no space in dir
201             jmp error_exit
```

→

```
202 prep8:      mov ax,0
203             call move_entry         ;copy entry 0 into first availability slot
204 prep9:      mov si,DIR_data         ;check second entry for availability
205             add si,32               ;point SI to second entry
206             cmp byte ptr [si],0     ;first byte zero?
207             je prep13               ;yes, then it's available
208             cmp byte ptr [si],0E5h  ;erased file?
209             jne prep9a              ;no, then move it
210             mov byte ptr [si],0     ;mark entry
211             jmp prep13              ;continue
212 prep9a:     mov dx,2                ;not available
213             call find_entry
214             jnc prep10              ;continue if empty slot found
215             lea dx,errmsg5
216             jmp error_exit
217 prep10:     mov ax,1
218             call move_entry
219 ;
220 ;Copy each of the first 40 clusters currently in use to the clusters beyond the
221 ;first 40  and adjust the corresponding FAT entries accordingly.
222 ;
223 prep13:     mov cx,40               ;check 40 clusters
224             mov dx,2                ;start checking at cluter 2
225 prep14:     push cx                 ;save loop counter
226             call get_FAT_entry      ;get FAT entry current cluster
227             cmp ax,0                ;is it available?
228             je prep17               ;yes, then proceed to next cluster
229             cmp ax,0FF0h            ;is FAT entry less than FF0h?
230             jb prep14a              ;yes, then move cluster
231             cmp ax,0FF7h            ;is it greater than FF7h?
232             ja prep14a              ;yes, then move cluster
233             pop cx                  ;bad cluster - clean up stack
234             lea dx,errmsg9
235             jmp error_exit
236 prep14a:    mov FAT_entry,ax        ;save FAT entry for later
237             mov ax,last_cluster     ;get number of last available cluster
238             inc ax                  ;next one to egin search
239             mov bx,0
240             call find_cluster       ;find first cluster with FAT entry of zero
241             mov last_cluster,ax     ;store cluster number just found
242             call move_cluster       ;copy the old cluster to the new
243             jnc prep15              ;continue if no error in disk I/O
244             pop cx                  ;on error, clean up stack
245             jmp error_exit
246 prep15:     mov ax,FAT_entry        ;retrieve the FAT entry for the old cluster
247             push dx                 ;save the old cluster number
248             mov dx,last_cluster     ;target the new cluster
249             call put_FAT_entry      ;put the old FAT entry into the new
250             pop dx                  ;restore the old cluster number
251             mov ax,0                ;mark the old cluster available
252             call put_FAT_entry
253             mov bx,dx               ;get old cluster number to BX for search
```

→

```
254             mov ax,2                ;start search at cluster 2
255             call find_cluster       ;find FAT entry that referenced old cluster
256             jnc prep16              ;if found then continue
257             call search_DIR         ;if not, then look in the directory
258             jnc prep15a             ;if found, then continue
259             pop cx
260             lea dx,errmsg7
261             jmp error_exit
262 prep15a:    mov ax,last_cluster     ;get new cluster number
263             mov [bx],ax
264             jmp prep17
265 prep16:     push dx
266             mov dx,ax               ;change FAT entry that formerly reference
267             mov ax,last_cluster
268             call put_FAT_entry
269             pop dx
270 prep17:     pop cx
271             inc dx                  ;index to sequential cluster
272             loop prep14             ;loop until the first 40 cluster are clear
273 ;
274 ;Write two copies of the file allocation table and one copy of the disk
275 ;directory back to disk, then exit
276 ;
277 prep17a:    call write_FAT          ;write two copies of FAT
278             jnc prep18              ;continue if no error on write
279             lea dx,errmsg4
280             jmp error_exit
281 prep18:     call write_DIR          ;write the directory to disk
282             jnc prep19
283             lea dx,errmsg4
284             jmp error_exit
285 prep19:     lea dx,msg1
286             mov ah,9
287             int 21h
288             mov ax,4C00h            ;terminate program with return code 0
289             int 21h
290 prep        endp
291 ;
292 ;-----------------------------------------------------------------------
293 ; GET_FAT_ENTRY routine returns the FAT entry for the indicated cluster.
294 ; Entry : DX - cluster number (2-XXX)
295 ; Exit  : AX - FAT entry
296 ;-----------------------------------------------------------------------
297 get_FAT_entry proc near
298             mov ax,3                ;multiply cluster number by 3
299             push dx                 ;save cluster number
300             mul dx                  ;multiply
301             pop dx                  ;restore cluster number
302             shr ax,1                ;divide by 2 to obtain offset into FAT
303             mov bx,ax               ;transfer to BX
304             add bx,FAT_data         ;make address relative to code segment
305             mov ax,[bx]             ;get word
306             test dx,1               ;is cluster number odd or even?
307             je getf1                ;even, then jump
308             mov cl,4                ;odd, then keep upper 12 bits of word
309             shr ax,cl               ;by shifting right 4 bits
310             ret
311 getf1:      and ax,0FFFh            ;even number, then mask off upper 4 bits
312             ret
313 get_FAT_entry endp
314 ;
315 ;-----------------------------------------------------------------------
316 ; PUT_FAT_ENTRY places the designated number into the FAT entry for the named
317 ; cluster.
318 ; Entry : DX - cluster number (2-XXX)
319 ;         AX - entry
320 ;-----------------------------------------------------------------------
321 put_FAT_entry proc near
322             push ax                 ;save new FAT entry
323             push dx                 ;save cluster number
324             mov ax,3                ;multiply cluster by 3
325             mul dx
326             pop dx                  ;restore cluster number
327             shr ax,1                ;divide by 2
328             mov bx,ax               ;transfer to BX
329             add bx,FAT_data         ;complete offset address of FAT entry
330             pop ax                  ;restore FAT entry
331             test dx,1               ;is cluster odd or even?
332             je putf1                ;even, then jump
333             mov cl,4                ;odd - shift entry 4 bits left
334             shl ax,cl
335             and word ptr [bx],0Fh   ;zero upper 12 bits of word
336             or [bx],ax              ;install new FAT entry
337             ret
338 putf1:      and [bx],0F000h         ;even - zero lower 12 bits
339             or [bx],ax              ;place entry in lower 12 bits
340             ret
341 put_FAT_entry endp
342 ;
343 ;-----------------------------------------------------------------------
344 ; The following four  routines write and read complete copies of the FAT and
345 ; the disk directory to and from the selected drive.
346 ;-----------------------------------------------------------------------
347 read_FAT    proc near               ;read one copy of the FAT into memory
348             mov al,drive            ;define drive
349             mov dx,1                ;start at sector 1
350             mov cx,FAT_sectors      ;set number of sectors
351             mov bx,FAT_data         ;define buffer address
352             int 25h                 ;DOS absolute sector read
353             pop dx                  ;clean flag of stack
354             ret
355 read_FAT    endp
356 ;
357 read_DIR    proc near               ;read one copy of directory into memory
```

```
358             mov al,drive            ;define drive
359             mov dx,FAT_sectors      ;get number of sectors per FAT
360             shl dx,1                ;double it for two copies of the FAT
361             inc dx                  ;directory starts in next sector
362             mov cx,DIR_sectors      ;number of sectors to read
363             mov bx,DIR_data         ;address of storage buffer
364             int 25h                 ;DOS absolute sector read
365             pop dx                  ;clean flag of stack
366             ret
367 read_DIR    endp
368 ;
369 write_FAT   proc near              ;write two copies of the FAT to disk
370             mov al,drive            ;define drive
371             mov dx,1                ;begin at sector 1
372             mov cx,FAT_sectors      ;get number of sectors per FAT
373             mov bx,FAT_data         ;define buffer address
374             int 26h                 ;DOS absolute sector write
375             pop dx                  ;clean flag of stack
376             jc wfat1                ;exit on write error
377             mov al,drive            ;repeat for second copy of FAT
378             mov dx,1
379             mov cx,FAT_sectors
380             add dx,cx
381             mov bx,FAT_data
382             int 26h
383             pop dx
384 wfat1:      ret
385 write_FAT   endp
386 ;
387 write_DIR   proc near              ;write disk directory image in memory to disk
388             mov al,drive            ;define drive
389             mov dx,FAT_sectors      ;get number of sectors per FAT
390             shl dx,1                ;double it for two copies
391             inc dx                  ;directory starts in next sector
392             mov cx,DIR_sectors      ;number of sectors to read
393             mov bx,DIR_data         ;address of data to be written
394             int 26h                 ;DOS absolute sector write
395             pop dx                  ;clean flag of stack
396             ret
397 write_DIR   endp
398 ;
399 ;-------------------------------------------------------------------------
400 ; FIND_ENTRY searches thru the image of the disk directory stored in memory
401 ; and finds first available directory entry.
402 ; Entry : DX - starting entry (0-XXX)
403 ; Exit  : DX - first available entry
404 ;         CF - clear : available entry found
405 ;              set   : not found
406 ;-------------------------------------------------------------------------
407 find_entry  proc near
408             mov cx,DIR_entries      ;set maximum number of tries in CX
409             sub cx,dx
```

→

```
410             push cx                 ;save count value
411             mov bx,dx               ;get starting entry number in BX
412             mov cl,5                ;multiply BX by 32
413             shl bx,cl
414             add bx,DIR_data         ;complete offset address
415             pop cx                  ;restore loop count
416 findent1:   cmp byte ptr [bx],0     ;is the first byte zero?
417             je findent2             ;yes, then it's available
418             cmp byte ptr [bx],0E5h  ;is it erased file?
419             je findent2             ;yes, then it's available
420             inc dx                  ;not available - check next entry
421             add bx,32               ;address of next entry
422             loop findent1           ;loop back if some remain
423             stc                     ;none available - set CF
424             ret
425 findent2:   clc                     ;entry found - clear CF
426             ret
427 find_entry  endp
428 ;
429 ;-------------------------------------------------------------------------
430 ; MOVE_ENTRY routine transfers a directory entry from one space to another
431 ; and sets the old entry to indicate that it's available.
432 ; Entry : AX - source entry (0-XXX)
433 ;         DX - target entry (0-XXX)
434 ;-------------------------------------------------------------------------
435 move_entry  proc near
436             mov si,ax               ;convert entry number to offset address in SI
437             mov cl,5
438             shl si,cl
439             add si,DIR_data
440             mov di,dx
441             mov cl,5
442             shl di,cl
443             add di,DIR_data
444             push si                 ;save address of old entry for late
445             mov cx,32               ;32 bytes per entry
446             rep movsb               ;move all 32 bytes
447             pop si                  ;get address of old entry
448             mov byte ptr [si],0     ;mark entry as 'Never used'
449             ret
450 move_entry  endp
451 ;
452 ;-------------------------------------------------------------------------
453 ; FIND_CLUSTER routine finds the first cluster whose FAT entry matches the
454 ; number specified in BX.
455 ; Entry : AX - starting cluster (2-XXX)
456 ;         BX - FAT entry
457 ; Exit  : AX - first cluster found
458 ;         CF - clear : cluster found
459 ;              set   : not found
460 ;-------------------------------------------------------------------------
461 find_cluster  proc near
```

→

```
                mov cx,clusters          ;get number of clusters
                add cx,2                 ;calculate max number of search loops
                sub cx,ax
                push dx                  ;save DX
                mov dx,ax                ;starting cluster in DX
findcls1:       push cx                  ;save count of cluster s checked
                push bx                  ;save FAT entry value
                call get_FAT_entry       ;get FAT entry for designated cluster
                pop bx                   ;restore FAT entry to BX
                pop cx                   ;restore count
                cmp ax,bx                ;is it what we're searching for?
                je findcls2              ;yes, then terminate search
                inc dx                   ;no, then try next cluster
                loop findcls1
                pop dx                   ;restore DX
                stc                      ;set CF to indicate cluster not found
                ret
findcls2:       mov ax,dx                ;put cluster number in AX
                pop dx                   ;restore DX
                clc                      ;clear CF to indicate success
                ret
find_cluster    endp
;
;-----------------------------------------------------------------------------
; CLS2SEC routine returns the sector number that corresponds to the cluster
; number input.
; Entry : DX - cluster number (2-XXX)
; EXit  : DX - sector number
;-----------------------------------------------------------------------------
cls2sec         proc near
                sub dx,2                 ;subtract two from the cluster number
                mov ax,sec_per_cluster   ;get number of sectors in each cluster
                mul dx                   ;multiply DX by sectors in each cluster
                mov dx,ax                ;transfer to DX
                mov ax,FAT_sectors       ;get number of sectors in FAT
                shl ax,1                 ;double it for two copies of the FAT
                add dx,ax                ;add FART sector
                add dx,DIR_sectors       ;add directory sectors
                inc dx                   ;data starts in next sector
                ret
cls2sec         endp
;
;-----------------------------------------------------------------------------
; MOVE_CLUSTER routine copies the contents of the source cluster into the
; target cluster.
; Entry : DX - source cluster (2-XXX)
;         AX - target cluster (2-XXX)
; Exit  : CF - clear : no error in disk I/O
;              set   : I/O error
;-----------------------------------------------------------------------------
move_cluster    proc near
                push dx
                push ax
                call cls2sec             ;get sector number of source cluster
                mov cx,sec_per_cluster   ;number of sector to read
                mov al,drive             ;specify drive
                mov bx,cbuffer           ;set data transfer area
                int 25h                  ;DOS sector read
                pop ax
                pop dx
                jc movec1                ;exit on error
                call cls2sec             ;get sector number of target cluster
                mov cx,sec_per_cluster   ;number of sector to read
                mov al,drive             ;specify drive
                mov bx,cbuffer           ;set data transfer area
                int 26h                  ;DOS sector write
                pop ax
                pop dx
                jc movec2
                ret
movec1:         pop dx
                lea dx,errmsg3           ;'Read Error' message
                ret
movec2:         lea dx,errmsg4           ;'Write Error' message
                ret
move_cluster    endp
;
;-----------------------------------------------------------------------------
; SEARCH_DIR routine searches each entry in the disk directory image for a
; starting cluster entry that matches the one input in DX.
; Entry : DX - cluster number (2-XXX)
; Exit  : BX - address of starting cluster word
;         CF - clear : match found
;              set   : not found
;-----------------------------------------------------------------------------
search_DIR      proc near
                mov bx,DIR_data          ;get starting address of directory image
                add bx,90                ;starting cluster offset - 3rd directory entry
                mov cx,DIR_entries       ;get maximum number of search loops
                sub cx,2                 ;skip first two entries
sdir1:          cmp [bx],dx              ;do the words match for this entry?
                je sdir2                 ;yes, then clear CF and exit
                add bx,32                ;no, then adjust BX for next entry
                loop sdir1               ;try next entry
                stc                      ;no match found
                ret
sdir2:          clc                      ;match found
                ret
search_DIR      endp
;
endprog         label byte               ;start of buffer area
;
code            ends
                end begin
```

เป็น 2 ชุด ชุดละ 2 เซกเตอร์) และข้อมูลทั่วไปของผู้ใช้ 708 เซกเตอร์

การจองเนื้อที่บนดิสค์นั้นดอสกำหนดให้จองได้ในหน่วยของคลัสเตอร์ โดยที่หนึ่งคลัสเตอร์จะประกอบด้วยหนึ่งหรือสองเซกเตอร์ก็ขึ้นกับฟอร์แมตของดิสค์ และดิสค์ซึ่งมีความจุ 360 กิโลไบต์ จะมี 354 คลัสเตอร์ (หมายเลข 2-355) สำหรับเก็บข้อมูล เมื่อดอสจะทำการเก็บไฟล์ใหม่ลงบนดิสค์ ดอสจะต้องตรวจหาคลัสเตอร์ว่างเพื่อจะเก็บไฟล์ใหม่นั้น และหากมีดอสจะเขียนข้อมูลบนดิสค์ทีละเซกเตอร์ นอกจากนั้นดอสยังต้องเก็บไดเรคทอรีสำหรับไฟล์นั้นด้วยโดยแต่ละไดเรคทอรีจะประกอบด้วย ชื่อไฟล์, แอตทริบิวของไฟล์, ขนาดไฟล์, วันที่และเวลาที่สร้างไฟล์ การเขียนข้อมูลของไฟล์นั้นดอสจะไม่เขียนบนเซกเตอร์ที่ต่อเนื่องกันจนจบไฟล์ แต่อาจแบ่งไฟล์ออกเป็นส่วน ๆ แล้วเขียนบนเซกเตอร์ซึ่งยังว่างอยู่ทั้งนี้ก็เพื่อที่จะได้ใช้พื้นที่บนดิสค์ได้อย่างมีประสิทธิภาพ

ในการอ่านไฟล์จากดิสค์นั้นดอสจำเป็นต้องรู้ว่าไฟล์นั้นเก็บอยู่ที่ใดบนดิสค์บ้าง ซึ่งดอสก็ทำได้โดยอ่านตำแหน่งที่เก็บของข้อมูลจาก FAT ที่ดอสเก็บไว้ขณะที่สร้างไฟล์ครั้งแรก FAT จะเป็นตารางที่เก็บสถานะของแต่ละคลัสเตอร์ไว้ จากตารางที่ 1 จะแสดงความหมายของค่าที่เก็บใน FAT เช่น ถ้าค่าที่เก็บใน FAT เป็น 000 แสดงว่าคลัสเตอร์นั้นยังว่างอยู่ สามารถนำมาใช้ชี้ตำแหน่งของข้อมูลที่จะเก็บบนดิสค์เป็นต้น

สมมติว่าดอสต้องการเก็บข้อมูลโดยใช้เนื้อที่ 3 คลัสเตอร์ และดอสพบว่าคลัสเตอร์หมายเลข 5, 10, 11 เป็นคลัสเตอร์ว่าง ขณะที่เขียนไฟล์ดอสเก็บหมายเลขคลัสเตอร์ 10 ไว้ในตาราง FAT ตำแหน่งที่ 5, เก็บหมายเลขคลัสเตอร์ 11 ไว้ในตาราง FAT ตำแหน่งที่ 10 และเก็บค่า FFFh ไว้ในตาราง FAT ตำแหน่งที่ 11 จะเห็นว่าจุดเริ่มต้นของไฟล์นี้อยู่ที่คลัสเตอร์หมายเลข 5 ซึ่งดอสจะต้องเก็บหมายเลขคลัสเตอร์เริ่มต้นนี้ไว้ในไฟล์ไดเรคทอรีที่ออฟเซต 26 ดังนั้นดอสก็จะสามารถเขียนและอ่านข้อมูลได้โดยใช้ตาราง FAT และไฟล์ไดเรคทอรี

การจัดเนื้อที่บนดิสค์ใหม่

เนื่องจากดอสมีข้อกำหนดว่าในดิสค์ที่มีซิสเต็มนั้นจะต้องมี IBMBIO.COM และ IBMDOS.COM ที่ซ่อนไว้ ซึ่งไฟล์ทั้งสองนี้ต้องเก็บไว้ในดิสค์ไดเรคทอรีสองตำแหน่งแรก นอกจากนี้ IBMBIO.COM ยังต้องเก็บในเซกเตอร์ที่อยู่ต่อเนื่องกันโดยเริ่มต้นจากคลัสเตอร์แรกของแผ่นด้วย

PSYS จะต้องย้ายข้อมูลในดิสค์ไดเรคทอรีสองตำแหน่งแรกออกก่อนให้เกิดที่ว่างเพื่อให้ SYS ทำงานได้โดยจะเรียก FIND_ENTRY มาค้นหาที่ว่าง จากนั้นจะเรียก MOVE_ENTRY มาย้ายข้อมูลไปไว้ที่ว่างนั้น เนื่องจากข้อจำกัดของ IBMBIO.COM ที่ต้องเริ่มต้นเก็บที่คลัสเตอร์แรกและต้องต่อเนื่องกันทั้งไฟล์ด้วย เราจึงจำเป็นต้องย้ายข้อมูลทั้งหมดที่อยู่ในบริเวณนี้ออกให้หมด โดยเราจะกันเนื้อที่ไว้ 40 คลัสเตอร์ ซึ่งเผื่อไว้สำหรับ DOS รุ่นใหม่ๆ ที่อาจมีขนาดไฟล์ใหญ่

การย้ายข้อมูลจะเริ่มต้นอ่านค่าจากตาราง FAT ของคลัสเตอร์หมายเลข 2 มาตรวจดูว่า ถ้ามีค่าอยู่ระหว่าง FF0h-FF7h แสดงว่าเป็นคลัสเตอร์เสีย หรือ ถ้ามีค่าเป็นศูนย์แสดงว่าเป็นคลัสเตอร์ว่าง แต่ถ้าเป็นค่าอื่นก็แสดงว่าคลัสเตอร์นี้มีข้อมูลเก็บอยู่ ดังนั้นเราจะต้องย้ายข้อมูลนี้ออก โดยจะเรียก FIND_CLUSTER ให้ค้นหาคลัสเตอร์ว่างที่ไม่อยู่ใน 40 คลัสเตอร์ที่เรากันไว้ จากนั้น MOVE_CLUSTER จะย้ายข้อมูลทั้งหมดไปไว้ที่คลัสเตอร์ว่างนั้น แล้วจะเอาค่าจากตาราง FAT ของคลัสเตอร์หมายเลข 2 มาเก็บไว้ใน FAT ในตำแหน่งของคลัสเตอร์ว่างนั้น จากนั้นก็จะเซตให้ FAT ของคลัสเตอร์หมายเลข 2 เป็นศูนย์

สุดท้ายเราจะต้องค้นหาว่ามีค่าใน FAT หรือ ในไฟล์ไดเรคทอรี (ออฟเซตที่ 26) ใดบ้างที่อ้างถึงคลัสเตอร์หมายเลข 2 ซึ่งเราจะต้องแก้ให้ค่าของ FAT นั้นชี้ไปที่คลัสเตอร์ว่างที่ FIND_CLUSTER ค้นพบได้แทน หากไม่มีข้อผิดพลาดใดเกิดขึ้นโปรแกรมก็จะกลับไปอ่านค่าจาก FAT ของคลัสเตอร์หมายเลขต่อไปมาจนครบ 40 คลัสเตอร์

ก่อนจบ

PSYS จะช่วยจัดเนื้อที่บนดิสค์ใหม่สำหรับให้ SYS ย้ายซิสเต็มไฟล์มายังดิสค์นั้น สำหรับข้อจำกัดในการใช้ PSYS ก็คือ PSYS จะทำงานกับฟลอปปี้ดิสค์เท่านั้น, ไม่ควรใช้ PSYS กับดิสค์ที่ป้องกันการก๊อบปี้ และดิสค์ที่มีไดเรคทอรีย่อย

ตารางที่ 1 แสดงค่าและความหมายที่ดอสเก็บใน FAT

ค่าใน FAT	ความหมายของดอส
000	คลัสเตอร์ว่างสำหรับเก็บไฟล์ใหม่
FF0-FF6	คลัสเตอร์ที่ดอสจองไว้
FF7	คลัสเตอร์เสีย
FFF	คลัสเตอร์สุดท้ายของไฟล์
อื่น ๆ	คลัสเตอร์ถูกใช้เก็บข้อมูลอยู่